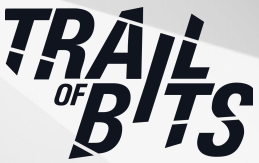

Offchain Labs Arbitrum Upgrades

Security Assessment (Summary Report)

January 16, 2024

Prepared for:
Harry Kalodner, Steven Goldfeder, and Ed Felten
Offchain Labs

Prepared by: **Troy Sargent and Kurt Willis**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

Mary O'Brien, Project Manager
mary.obrien@trailofbits.com@trailofbits.com

The following engineering director was associated with this project:

Josselin Feist, Engineering Director, Blockchain
josselin.feist@trailofbits.com

The following consultants were associated with this project:

Troy Sargent, Consultant
troy.sargent@trailofbits.com

Kurt Willis, Consultant
kurt.willis@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
January 2, 2024	Pre-project kickoff call
January 9, 2024	Report readout meeting and delivery of report draft
January 16, 2024	Delivery of comprehensive report

Executive Summary

Engagement Overview

Offchain Labs engaged Trail of Bits to review the security of Abritrum's security council elections upgrade and sequencer maximum verification time settings update,.

A team of two consultants conducted the review from January 2 to January 8, 2024, for a total of two engineer-weeks of effort. With full access to source code and documentation, we performed static and dynamic testing of the targets, using automated and manual processes.

Observations and Impact

We did not identify any significant issues in the changes to the governance codebase. However, we did identify several ways to improve documentation, adhere to best practices, and increase the reliability of tests.

Recommendations

Based on the findings identified during the security review, Trail of Bits recommends that Offchain Labs take the following step:

- **Remediate the findings disclosed in this report.** These findings should be addressed as part of a direct remediation or as part of any refactor that may occur when addressing other recommendations.

Finding Severities and Categories

The following tables provide the number of findings by severity and category.

EXPOSURE ANALYSIS

Severity	*Count*
High	0
Medium	0
Low	0
Informational	3
Undetermined	0

CATEGORY BREAKDOWN

Category	*Count*
Auditing and Logging	1
Undefined Behavior	2

Project Goals

The engagement was scoped to provide a security assessment of the Offchain Labs upgrades. Specifically, we sought to answer the following non-exhaustive list of questions:

- Do the upgrades perform the expected actions?
- Are there any unexpected side effects that are introduced by these updates?
- Do existing addresses match deployed addresses?
- Are the new configurations correct and sensible?
- Can the maintenance costs of the codebase be improved?
- Do the tests specify correctness and closely resemble the production environment?

Project Targets

The engagement involved a review and testing of the following targets.

governance PR#231

Repository	https://github.com/ArbitrumFoundation/governance
Version	PR#231 (`d50318e...5be7268`)
Type	Solidity
Platform	EVM

governance PR#233

Repository	https://github.com/ArbitrumFoundation/governance
Version	PR#233 (`4191773...242bc72`)
Type	Solidity
Platform	EVM

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches included the following:

- Manually reviewing the security council elections updates and the sequencer updates
- Reviewing the validity of hard-coded addresses, their deployed code, and their interfaces
- Reviewing the validity of hard-coded precompile addresses
- Checking whether new values in settings are sensible
- Reviewing documentation for any discrepancies

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	Discrepancy in comment about upgrade action	Auditing and Logging	**Informational**
2	Unresolved “TODO” comments in NomineeGovernorV2UpgradeActionTemplate	Undefined Behavior	**Informational**
3	Unnecessary duplication in inheritance	Undefined Behavior	**Informational**

Detailed Findings

1. Discrepancy in comment about upgrade action

Severity: **Informational**	Difficulty: **Low**
Type: Auditing and Logging	Finding ID: TOB-ARB-SCE-1
Target: `src/gov-action-contracts/AIPs/SetSeqMaxTimeVariation/AIPSetSequencerInboxMaxTimeVariationArbOneAction.sol`	

Description
The upgrade action to set the sequencer inbox maximum time variation contains an incorrect comment.

```
SetSequencerInboxMaxTimeVariationAction(
    ISequencerInboxGetter(0xd514C2b3aaBDBfa10800B9C96dc1eB25427520A0), // Arb One
Address Registry
    5760, // Delay blocks (same as current value)
    64, // New future blocks value
    86_400, //  Delay seconds (same as current value)
    768 // New future seconds value (delay blocks * 12)
)
```

Figure 1.1: Sequencer upgrade action (AIPSetSequencerInboxMaxTimeVariationArbOneAction.sol#13–19)

The comment for the new future seconds value (768) should read `(future blocks * 12)` instead of `(delay blocks * 12)`.

Recommendations
Short term, correct the comment in the upgrade action for both upgrade actions in Arbitrum One and Arbitrum Nova.

2. Unresolved “TODO” comments in NomineeGovernorV2UpgradeActionTemplate

Severity: **Informational**	Difficulty: **Medium**
Type: Undefined Behavior	Finding ID: TOB-ARB-SCE-2
Target: `src/gov-action-contracts/AIPs/NomineeGovernorV2UpgradeAction.sol`	

Description

The `NomineeGovernorV2` upgrade action contains unresolved “TODO” comments that need to be addressed before a final deployment and upgrade are possible.

```
constructor() NomineeGovernorV2UpgradeActionTemplate(
    0xdb216562328215E010F819B5aBe947bad4ca961e,
    0x8a1cDA8dee421cD060234706086059 34c16A05a0,
    address(0), // todo: new implementation
    50400,
    0x1D62fFeB72e4c360CcBbacf7c965153b00260417,
    0x0101010101010101010101010101010101010101010101010101010101010101 // todo: new
constitution hash
) {}
```

Figure 2.1: `NomineeGovernorV2` upgrade action (`NomineeGovernorV2UpgradeAction.sol`)

Recommendations

Short term, deploy the implementation contract, set the address in the constructor, and include the correct constitution hash.

3. Unnecessary duplication in inheritance

Severity: **Informational**	Difficulty: **Low**
Type: Undefined Behavior	Finding ID: TOB-ARB-SCE-3
Target: `governance/`	

Description

The `governance` codebase contains several instances of duplicate inheritance—a child contract inherits from a contract already in the inheritance tree of its parent. This may be necessary to control the C3 linearization of a contract's inheritance tree (but designs requiring this should generally be avoided). Since the following instances of redundant inheritance pertain only to virtual functions that are not overridden multiple times, the contracts that are repeatedly inherited can be removed (see appendix B).

```
Inheritance duplication in L1ArbitrumToken: Initializable (already inherited by ERC20Upgradeable)
Inheritance duplication in L1ArbitrumToken: ERC20Upgradeable (already inherited by
ERC20PermitUpgradeable)
Inheritance duplication in L2ArbitrumGovernor: Initializable (already inherited by
GovernorSettingsUpgradeable)
Inheritance duplication in L2ArbitrumGovernor: GovernorVotesUpgradeable (already inherited by
GovernorVotesQuorumFractionUpgradeable)
Inheritance duplication in L2ArbitrumToken: Initializable (already inherited by ERC20Upgradeable)
Inheritance duplication in L2ArbitrumToken: ERC20Upgradeable (already inherited by
ERC20BurnableUpgradeable)
Inheritance duplication in L2ArbitrumToken: ERC20PermitUpgradeable (already inherited by
ERC20VotesUpgradeable)
Inheritance duplication in UpgradeExecutor: Initializable (already inherited by
AccessControlUpgradeable)
Inheritance duplication in SecurityCouncilManager: Initializable (already inherited by
AccessControlUpgradeable)
Inheritance duplication in SecurityCouncilMemberElectionGovernor: Initializable (already inherited
by GovernorUpgradeable)
Inheritance duplication in SecurityCouncilMemberElectionGovernor: GovernorUpgradeable
(already inherited by GovernorVotesUpgradeable)
Inheritance duplication in SecurityCouncilMemberRemovalGovernor: Initializable (already
inherited by GovernorUpgradeable)
Inheritance duplication in SecurityCouncilMemberRemovalGovernor: GovernorUpgradeable
(already inherited by GovernorVotesUpgradeable)
Inheritance duplication in SecurityCouncilMemberRemovalGovernor: GovernorVotesUpgradeable
(already inherited by ArbitrumGovernorVotesQuorumFractionUpgradeable)
```

```
Inheritance duplication in SecurityCouncilNomineeElectionGovernor: Initializable (already
inherited by GovernorUpgradeable)
Inheritance duplication in SecurityCouncilNomineeElectionGovernor: GovernorUpgradeable
(already inherited by GovernorVotesUpgradeable)
Inheritance duplication in SecurityCouncilNomineeElectionGovernor: GovernorVotesUpgradeable
(already inherited by ArbitrumGovernorVotesQuorumFractionUpgradeable)
```

Figure 3.1: Terminal output from script in appendix B

Recommendations

Short term, remove redundant inheritance and thoroughly test for regressions in the storage layout of upgradeable contracts.

Long term, set and enforce coding standards pertaining to inheritance to avoid complexity and make future maintenance less burdensome.

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Detecting Duplicated Inheritance with Slither

The following script will discover redundant inheritance, as explained in TOB-ARB-SCE-3. Install Slither. Then run the script in the root of the repository with the command `python3 lint.py`.

```
from slither import Slither
from slither.utils.inheritance_analysis import detect_c3_function_shadowing

sl = Slither('.')

# Only analyze contracts that are not inherited by other contracts to prevent
recommending
# removing inheritance which a distinct child contract may depend on from the same
parent.
for contract in sl.contracts_derived:
    duplicated = set()
    for parent in contract.immediate_inheritance:
        for candidate in parent._inheritance:
            if candidate in contract.immediate_inheritance and candidate not in
duplicated:
                duplicated.add(candidate)
                print(f"Inheritance duplication in {contract.name}: {candidate.name}
(already inherited by {parent.name})\n")

    for dup in duplicated:
        for k,v in detect_c3_function_shadowing(contract).items():
            print("--WARNING-- Inheritance duplication affects C3 linearization:\n")
            print(k.canonical_name, ":\n")
            for func in v:
                print(func.canonical_name,"\n")
```

Figure B.1: Script to detect redundant inheritance